Grasp Planning

CS 6301 Special Topics:
Introduction to Robot Manipulation and Navigation

Outline and Learning Goals

Fundamental aspects of modeling a grasp

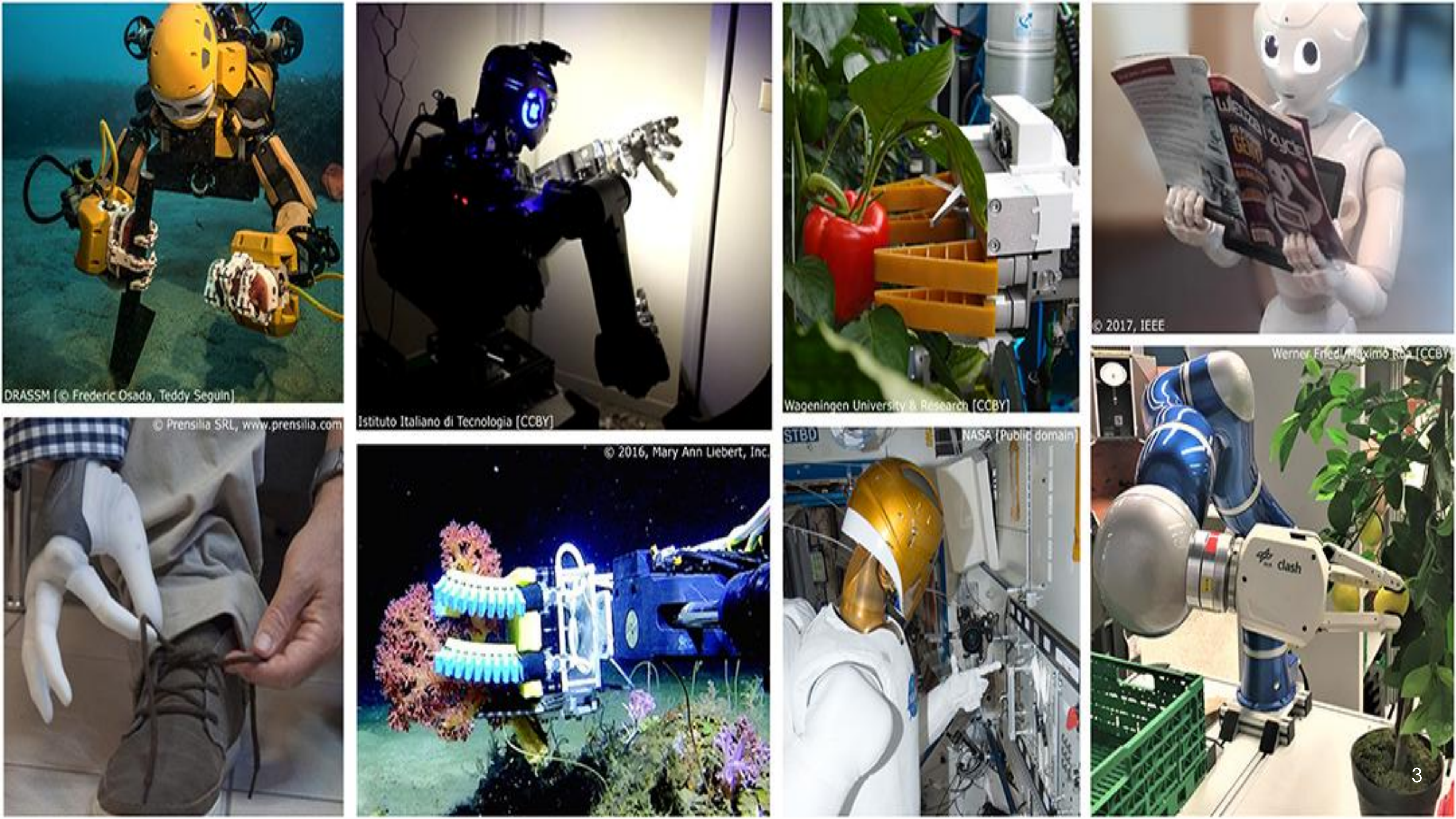
- Stability analysis

Overview of analytical contact models + Grasp Quality measures

- How to evaluate a grasp?

Learning-based and Data driven methods

- Review of some recent methods for grasp synthesis

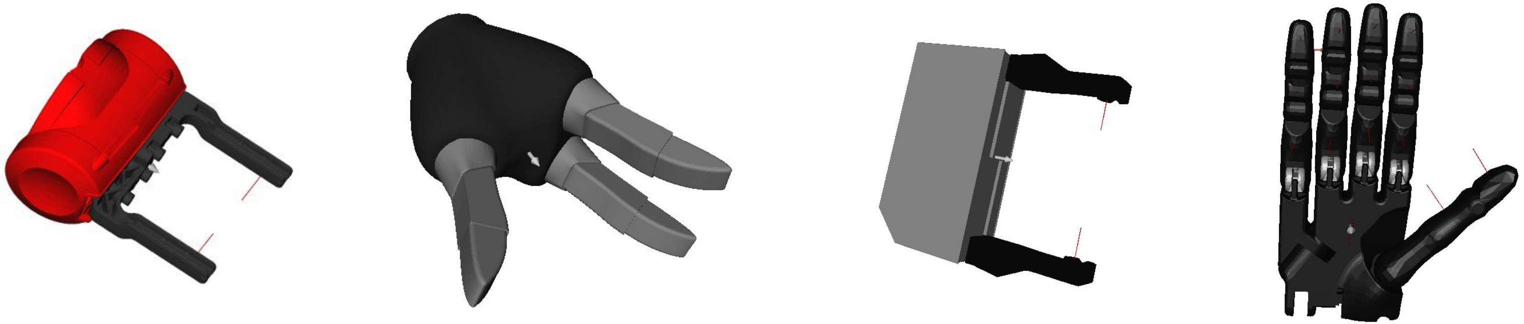

DRASSM [© Frederic Osada, Teddy Seguin]
Istituto Italiano di Tecnologia [CCBY]
Wageningen University & Research [CCBY]
© 2017, IEEE
© Prensilia SRL, www.prensilia.com
© 2016, Mary Ann Liebert, Inc.
NASA [Public domain]
STBD
Werner Friedl/Maximo Roa [CCBY]
clash

Introduction

Grasping is a part of a broader goal of robot manipulation:

1. Reaching the object: motion planning
2. Grasping the object:
3. Moving the grasped object: perhaps according some task

Different types of grippers (end effectors) involved: variation along kinematics.



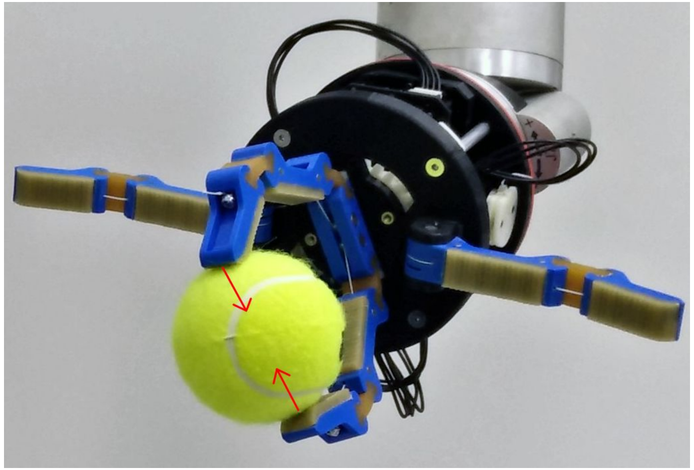
Gripper models loaded in the GraspIt simulator [3]

What does one mean by a Grasp?

Immobilize the object using the gripper fingers. Force/torque applied using the finger to constrain the object's motion

Representation:

- 6D Pose of Gripper Base
- N joint values (e.g. rotation angle) for all N joints in the gripper



Key Questions:

- Where to place the fingers (i.e contact points)
- Is the grasp any good? -- evaluating it.

Challenges

High dimensional problem

- Especially with Multi-fingered grippers
- 6D (Pose) + Joint configuration (2 to N)

Incomplete object observations

Object in a cluttered environment

Feasibility of motion plan

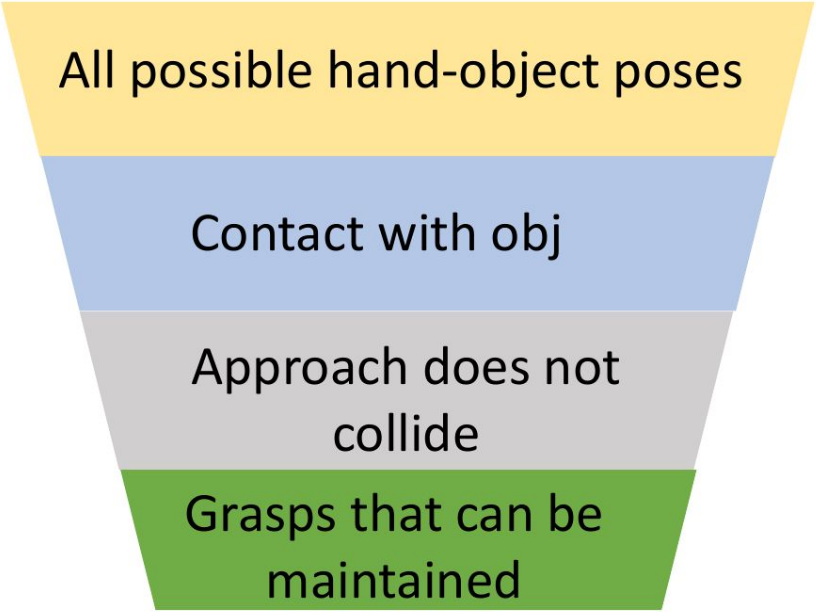
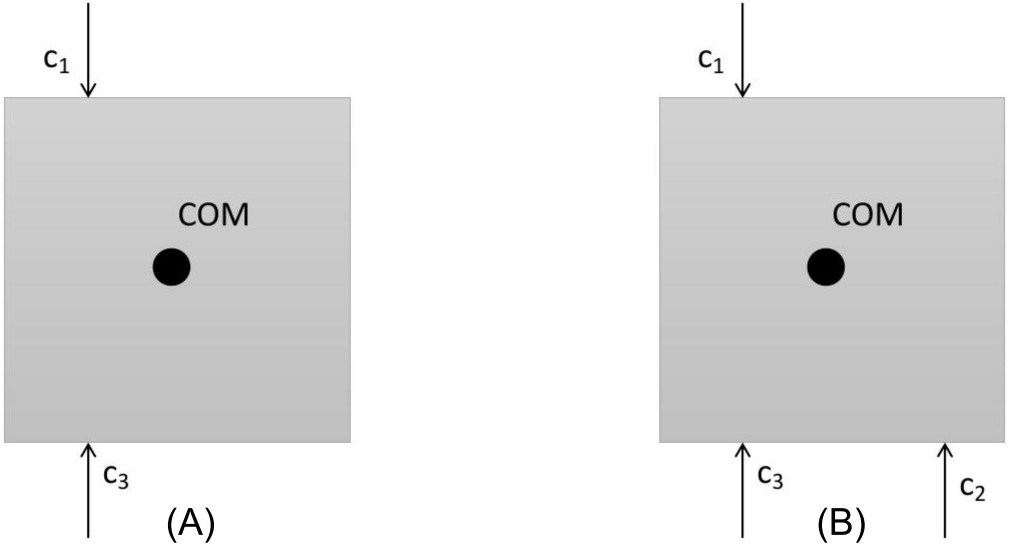


Image Credit: [1]

Grasp Contact Modeling

Goal: restrain object's motion via some force/torque through gripper finger

Simplified example of contacts: Can the object move in case A or B?



Credit: [1]

Modeling Assumptions

- Rigid Bodies
- We can apply unlimited force
- Contact Type: Point on a Plane
- Presence of friction between object’s surface and finger
- Hard-finger contact:
  - Small contact patch (point-based)
  - No moment due to friction, only friction force considered
- Contact points are already known i.e finger locations on object

Start out with modeling a *single* point of contact

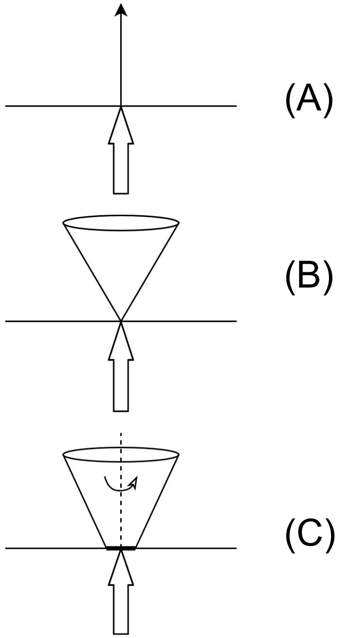
Point Contact Models

Different types of point-plane contact models

A. Frictionless point contact: only normal force

B. **Point contact with friction**: hard finger

C. Soft-finger: we additionally get a moment due to the friction



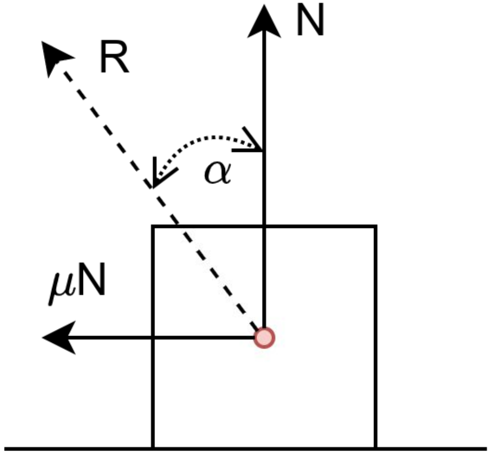
Each model will be encoded into a matrix **G** we will see later

Friction Cone

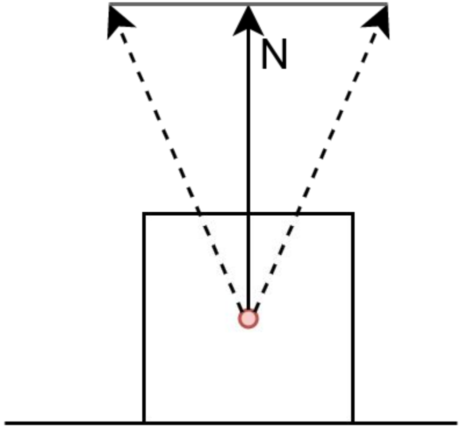
Assume friction on the object's surface with friction coefficient μ

In limiting friction: hence, force from friction Ft <= μ * N

N = Normal force from the surface (shown at COM for convenience)



$$\alpha = \tan^{-1}(\mu)$$



Friction Cone

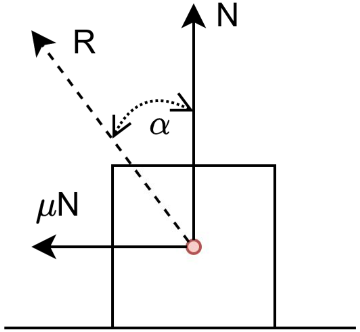
Due to friction, we not only have normal contact force N, but also some *tangential force*

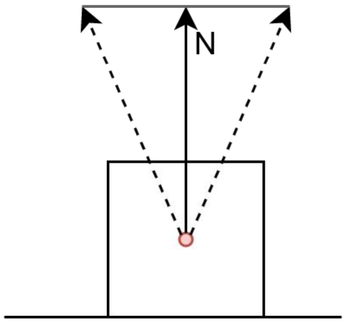
Orthogonal to the Normal force vector

Alpha: the maximum angle any resultant force R can make before slipping occurs (**angle of friction**)

The *Cone* is the space of all **valid resultant forces**

Non-limiting friction => angle will be less than alpha!





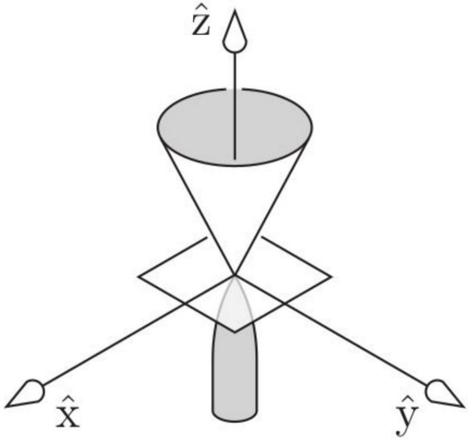
$$\alpha = \tan^{-1}(\mu)$$

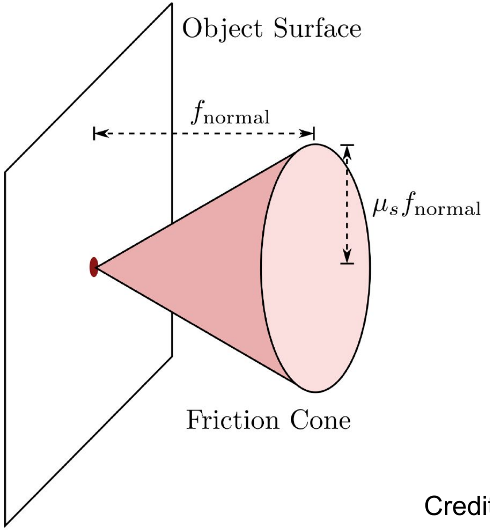
Friction Cone in 3D

Similarly in 3D, we get an actual Cone for the space of all possible force resulting from a contact point

Friction Cone for a specific contact point is the set **F** = { (fx, fy, fz) } satisfying the equation:

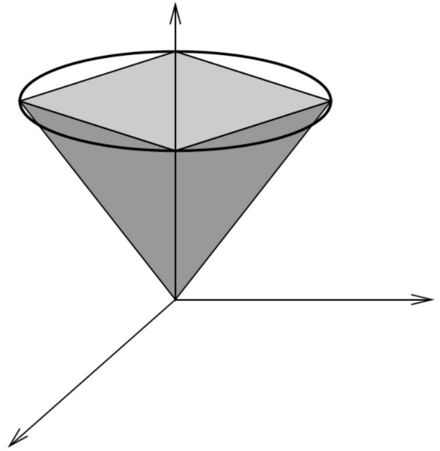
$$\sqrt{f_x^2 + f_y^2} \leq \mu \, f_z \quad , \; f_z \geq 0$$





Credit: [1]

Approximating the Friction Cone



In practice, we can approximate the friction cone by a set of vectors

One can approximate using *m* vectors **f_**(j) lying on the cone

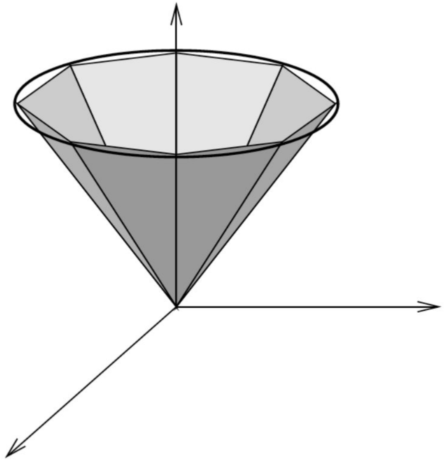
$$f_j^{(x)} = \mu \ \cos(\frac{2\pi j}{m})$$

$$f_j^{(y)} = \mu \ \sin(\frac{2\pi j}{m})$$

Assumed that f_z = 1 for convenience. We can apply scaling factor if needed.

$$f_j^{(z)} = 1$$

$$f_j \ ; \ j = 1, \ldots, m$$



Grasp Wrench

Recall, Wrench = 6D quantity that represents the forces+torques acting on a body

Each contact point applies a wrench on the body

For the contact force **f**, the **torque** = **d** X **f** (d = distance to object’s center of mass)

Note: We analyzed the cone in a *local contact point frame*

Assume **G** to be a matrix that models the 1) **Contact model** 2) **Transformation from local contact frame to object centric frame**

Wrench_i = G_i * f_i [For the i-th contact point]

Total Wrench = Sum(i=1,..,k) Wrench_i

$$\boldsymbol{w} = \sum_{i=1}^{k} G_i \boldsymbol{f}_i$$

Credit: [1]

Grasp Wrench Space (GWS)

For each i-th contact point, we have its friction cone $\mathcal{F}_i$

The grasp wrench space W for a grasp with k contact points is the set of all possible wrenches **w** that can be applied to the object through admissible forces

Note: The space is over all *permissible* force vectors f_i

Large grasp wrench space => Can compensate for bigger set of external forces!

GWS is 6D for 3D objects and 3D for 2D objects

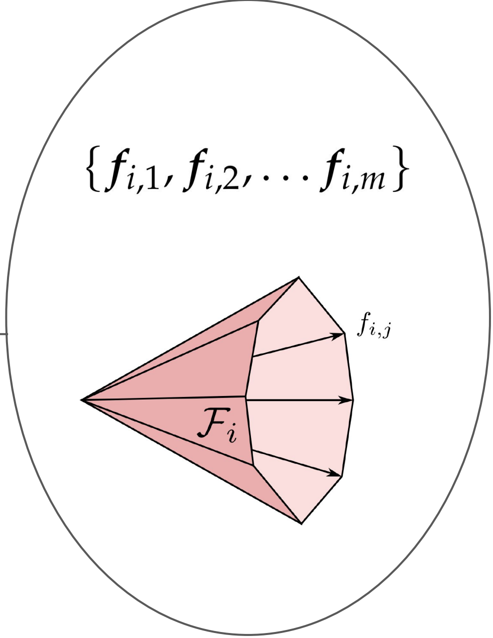
$$\mathcal{W} := \{\boldsymbol{w} \mid \boldsymbol{w} = \sum_{i=1}^{k} G_i \boldsymbol{f}_i, \quad \boldsymbol{f}_i \in \mathcal{F}_i, \quad i = 1, \ldots, k\}$$

Credit: [1]

Computing Grasp Wrench Space

Computing GWS looks difficult! Many, many combinations!

Key: Use the *discretized* friction cone to represent **any** permissible force f_i as a **linear, positive combination**



$$f_i = \sum_{j=1}^{m} \alpha_{i,j} f_{i,j}, \quad \alpha_{i,j} \geq 0$$

Also constrain value of each coefficient as we cannot have very large forces in practice!

If coefs sum bounded by 1 => Cone is approximated by **Convex Hull** of $\{f_{i,1}, f_{i,2}, \dots f_{i,m}\}$

$$\sum_{j=1}^{m} \alpha_{i,j} \leq 1$$

Credit: [1]

Computing Grasp Wrench Space

Hence by using the approximation to the friction cone, the GWS looks as follows

K is the number of contact points

M is the number of force approximating each friction cone for i-th contact point

$$\mathcal{W} = \{\boldsymbol{w} \mid \boldsymbol{w} = \sum_{i=1}^{k} \boldsymbol{w}_i, \quad \boldsymbol{w}_i = \sum_{j=1}^{m} \alpha_{i,j} \boldsymbol{w}_{i,j}, \quad \boldsymbol{w}_{i,j} = \begin{bmatrix} \boldsymbol{f}_{i,j} \\ \lambda(\boldsymbol{d}_i \times \boldsymbol{f}_{i,j}) \end{bmatrix}$$

$$\sum_{j=1}^{m} \alpha_{i,j} \leq 1, \quad \alpha_{i,j} \geq 0\}.$$

This is true for **all** i = 1,...,K

Credit: [1]

Recap

K is the number of contact points

M is the number of force approximating each friction cone for i-th contact point

1. Compute the approximate friction cone for *each* contact point
   a. Here we compute Convex Hull**s** for each contact's cone
2. For each force f_i on i-th contact, break it down using the approximate friction cone vectors
   a. Represent using the coefficients alphas
3. Can use this breakdown in computing GWS

Credit: [1]

Grasp Closure: Force & Form

Force Closure Grasp:

- If for any *external* wrench, there exist contact wrenches that can cancel it
- Can resist the external wrench by pushing more firmly.

Form Closure Grasp:

- Object is kinematically constrained and cannot move (even with no application of forces / regardless of surface friction).
- Locked Joint angles for the gripper
- Example: *enveloping an object* -- cannot move in any direction!

Credit: [1]

Force Closure Grasps

To resist any external wrench **w** means cancelling it out via the K contact forces

$$-\boldsymbol{w} = \sum_{i=1}^{k} G_i \boldsymbol{f}_i. \quad \boldsymbol{f}_i \in \mathcal{F}_i, \quad i = 1, \ldots, k$$

Theorem: To resist arbitrary external wrenches, GWS must **contain the origin** in its interior

Good way to check if a Grasp is in Force Closure or not!

But need to compute GWS first to check for origin in interior

Credit: [1]

Grasp Wrench Hull (GWH)

Computing the GWS even with the friction cone approximation can be difficult!

Modify the constraint on the coefficients alpha_ij: make it a **convex combination**

Note that sum over the alphas is **now equal to 1** (rather than <= 1)

$$\tilde{\mathcal{W}} = \{\boldsymbol{w} \mid \boldsymbol{w} = \sum_{i=1}^{k}\sum_{j=1}^{m}\alpha_{i,j}\boldsymbol{w}_{i,j}, \quad \boldsymbol{w}_{i,j} = \begin{bmatrix} \boldsymbol{f}_{i,j} \\ \lambda(\boldsymbol{d}_i \times \boldsymbol{f}_{i,j}) \end{bmatrix}, \quad \sum_{i=1}^{k}\sum_{j=1}^{m}\alpha_{i,j} = 1, \quad \alpha_{i,j} \geq 0\}$$

GWH is the Convex Hull of all individual wrenches w_ij

Key point: **GWH is subset of GWS** => Check for presence of origin inside GWH!

Credit: [1]

Recap

1. Grasp = K contact points with forces
2. Use *Origin inside interior of GWS* test to check for Force Closure of Grasp
3. GWS is still difficult to compute!
4. But need to compute it to test if origin lies inside or not
5. Compute Grasp Wrench Hull = Convex Hull of individual wrenches of friction cone approximating vectors
6. As GWH <= GWS (subset), if origin inside GWH => origin inside GWS!

Credit: [1]

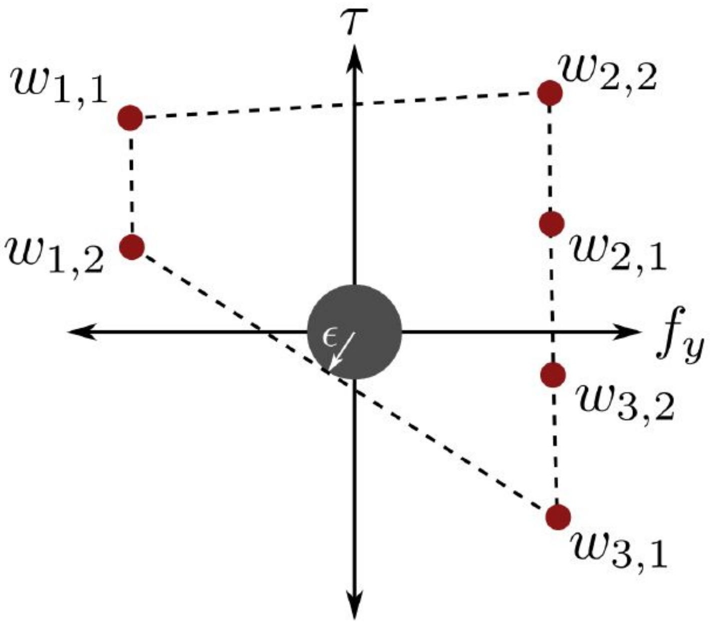
Grasp Quality

Often we may need to go beyond simple force closure. Comparing two force closure grasps → which one is better?

Use metrics based on Grasp Wrench Hull:

(a) **Epsilon**

(b) **Volume**



Credit: [1]

Grasp Quality

Epsilon: Worst Case Metric!

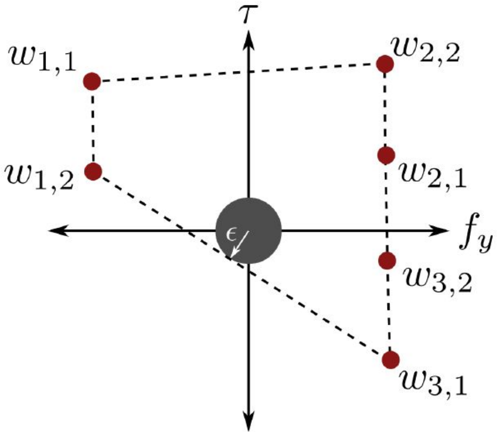
= radius of largest ball (in 6D) that is centered at origin and contained within the Grasp Wrench Hull

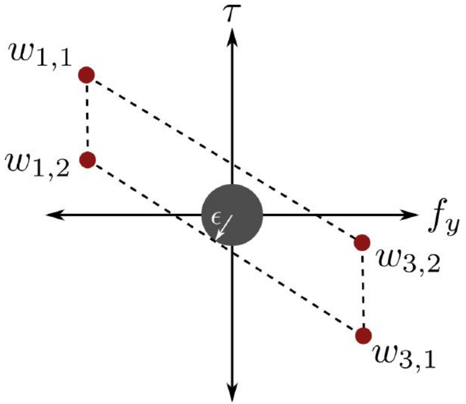
= Magnitude of *smallest* external wrench that pushes grasp to its limits

Volume: compute the (6D) volume of GWH

This quality considers the **average case**!

Can used to select grasps with similar epsilon





Credit: [1]

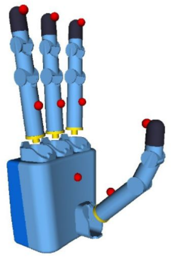
GraspIt Simulator Overview

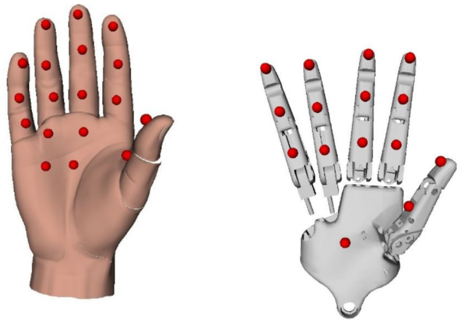
Graspit poses grasp planning as optimization problem

E is an objective function = f(**p**, **t**)

p is the pose and **t** is the gripper joint configuration.

- Simulated Annealing is used to minimize **E**
- **E** is an energy function computed using pre-defined contact points on the gripper
    - Sum of distances between gripper contact points and object
    - Angular differences between the surface normal at contact point and closest object point
- Evaluate intermediate steps using Grasp Quality measures





Problems with such modeling

Lot of assumptions may be invalid in real world grasping!

- Coefficient of friction between gripper finger and object surface = ???
- Complete object model along with its center of mass = ???
- Object Pose may be unknown
- Rigid vs deformable bodies
- Real time computation constraints!

Learning based methods

Graspit requires access to the object’s full 3D model => Infeasible in practice!

Alternative:

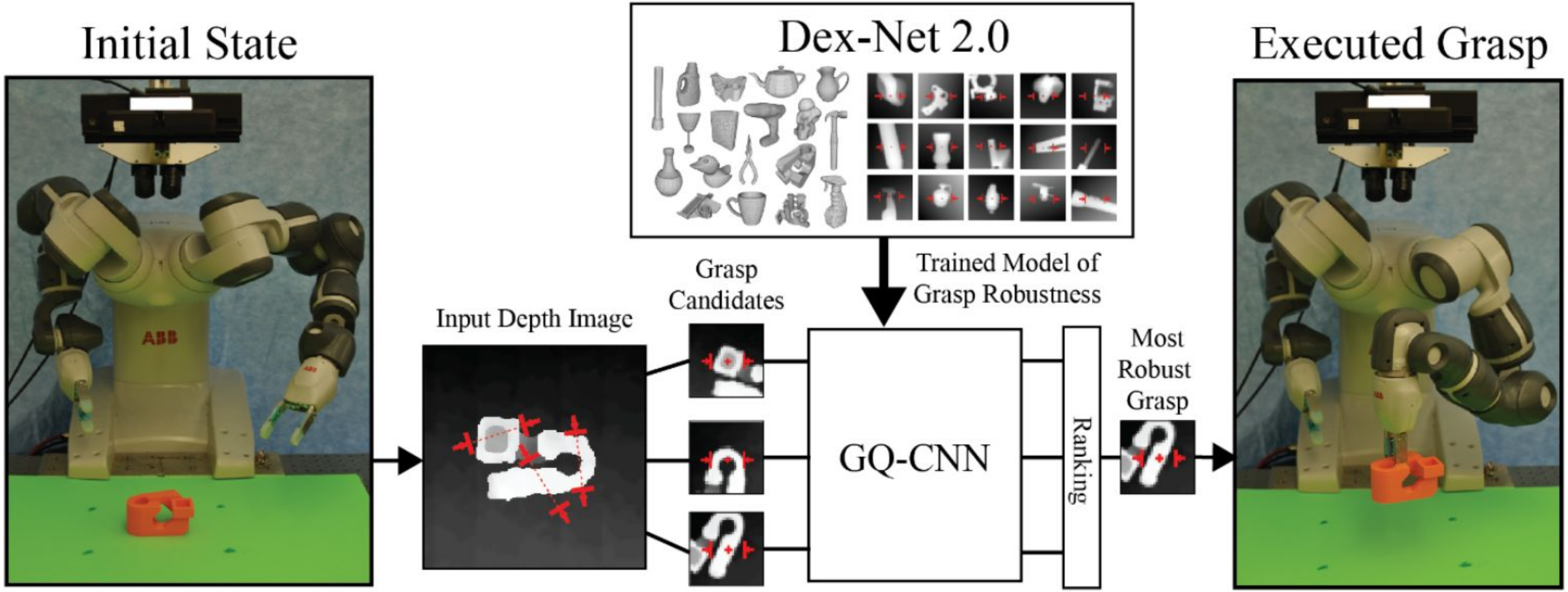
Learn mapping from images/point clouds to grasps

- Utilize dataset of synthetic grasps (generated using previous methods)
- Train NNs to evaluate the grasps

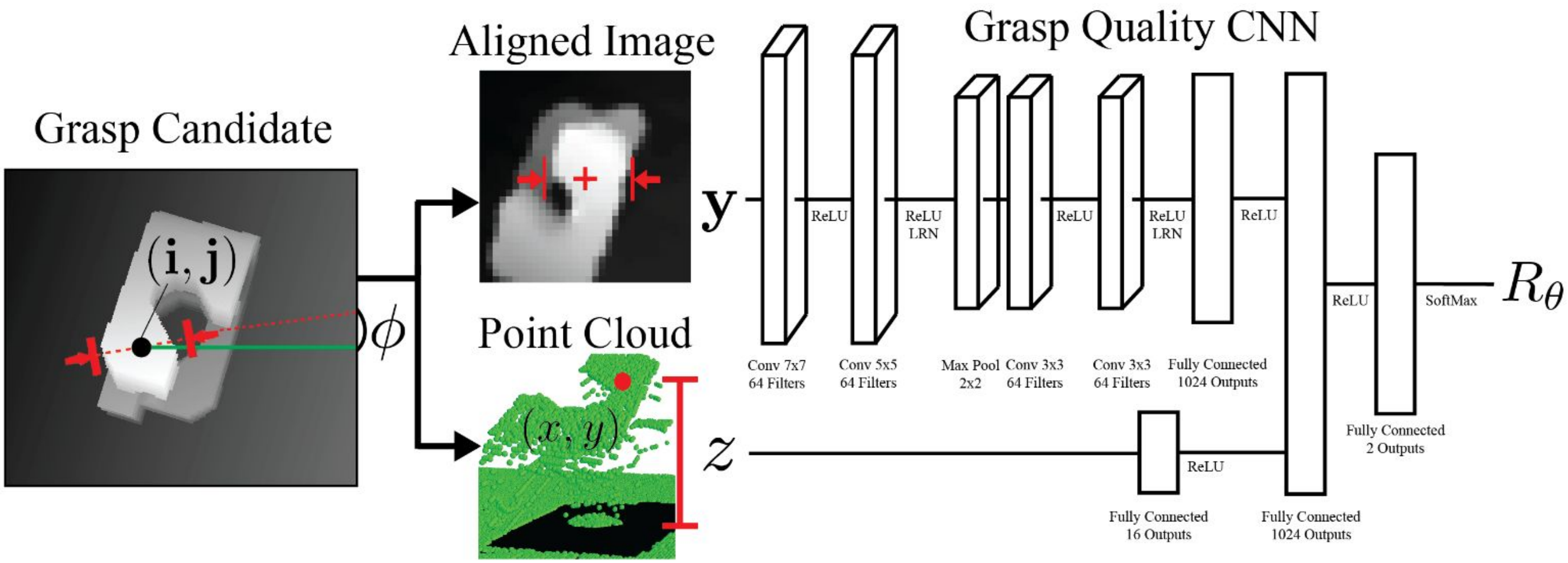
Example: Given a pixel (u,v) on the image, is it a valid grasping point (1) or not (0)

One Approach: Dex-Net 2.0 [4]

Dex-Net 2.0


Initial State
Dex-Net 2.0
Executed Grasp
Trained Model of
Grasp Robustness
Grasp
Candidates
Input Depth Image
GQ-CNN
Ranking
Most
Robust
Grasp

Dex-Net 2.0



Grasp Candidate
(i, j)
φ
Aligned Image
y
Point Cloud
(x, y)
z
Grasp Quality CNN
Conv 7x7
64 Filters
ReLU
Conv 5x5
64 Filters
ReLU
LRN
Max Pool
2x2
Conv 3x3
64 Filters
ReLU
Conv 3x3
64 Filters
ReLU
LRN
Fully Connected
1024 Outputs
ReLU
Fully Connected
16 Outputs
ReLU
Fully Connected
1024 Outputs
ReLU
Fully Connected
2 Outputs
SoftMax
R_θ

Beyond Planar Grasping: 6-DoF Grasping

In DexNet, grasps are represented by oriented rectangles on a 2D image

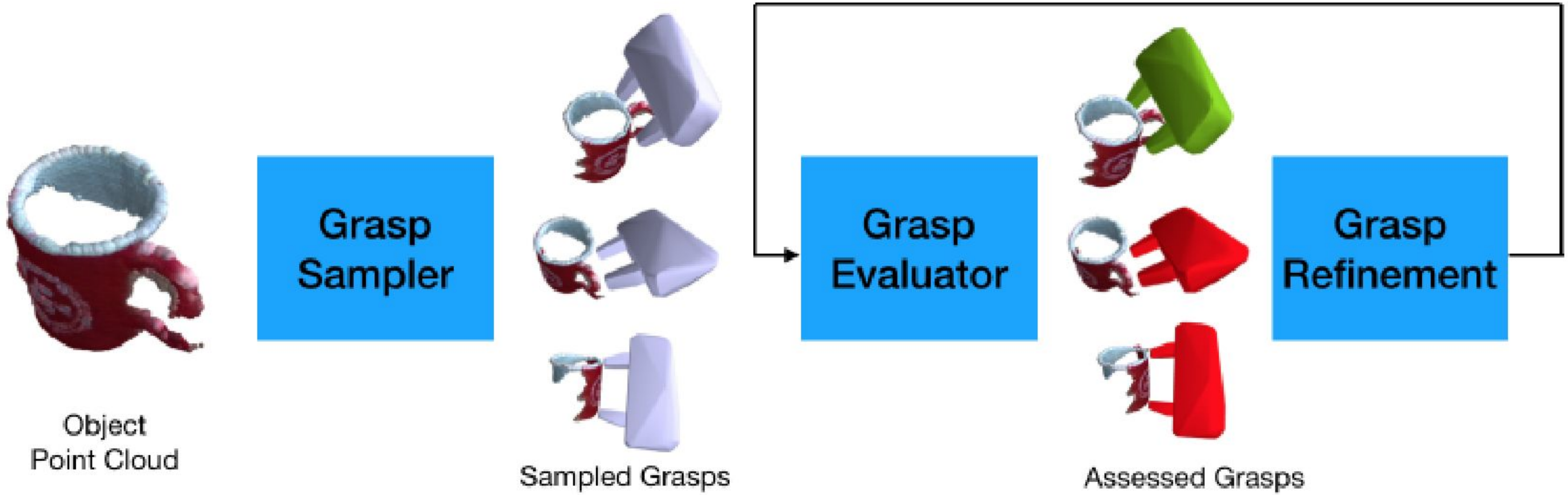
Only anti-podal points considered for grasp candidates

But this does not make full use of robot joints

Grasp pose is actually in 6D (3D rotation + translation)

One Approach: 6-DoF GraspNet [5]

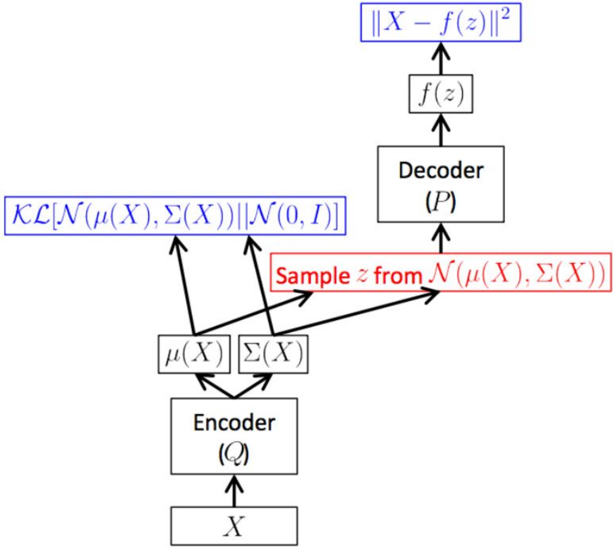
6-DoF GraspNet

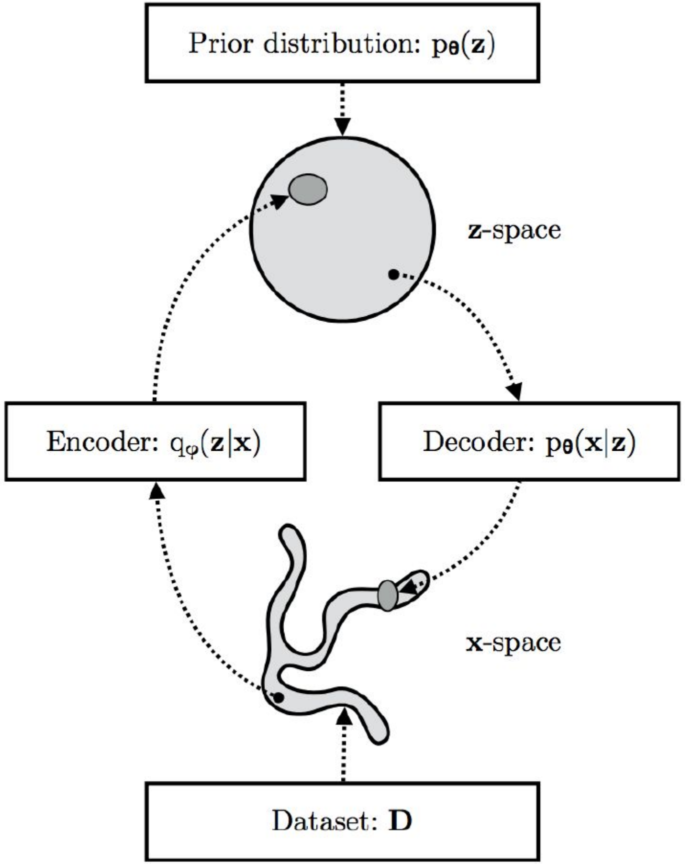


Goal

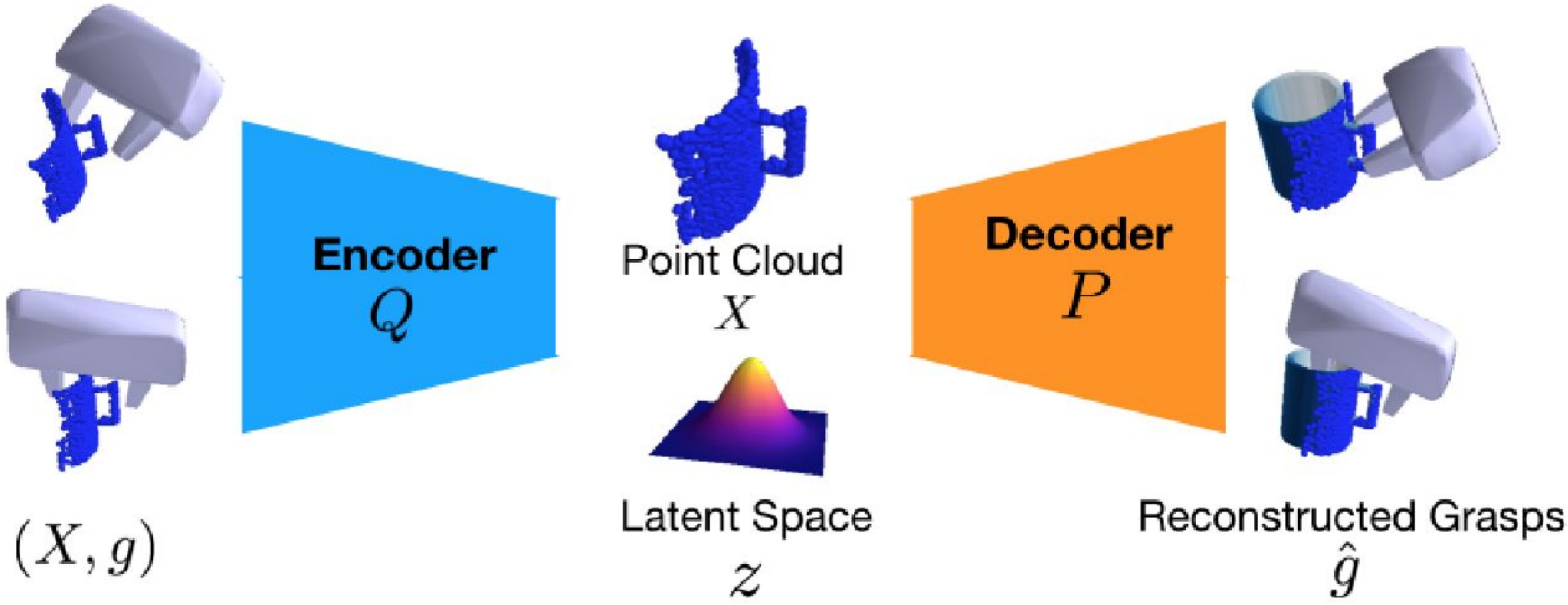
Objective: Having a generator model that samples from the distribution of the data

Find *likely* z's (given X) using Q(z|X) → encoder!





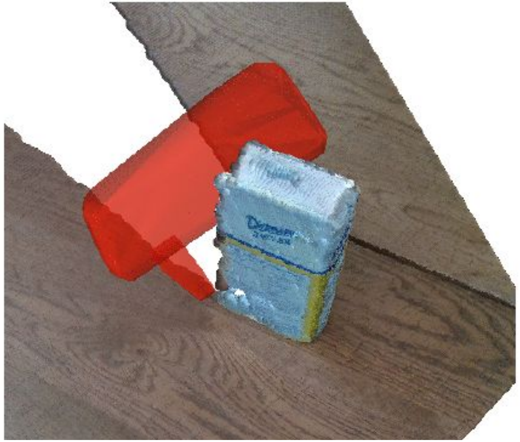
6-DoF GraspNet

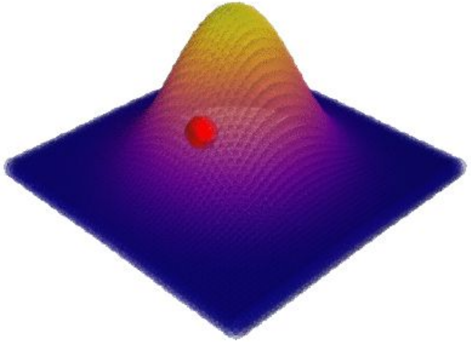


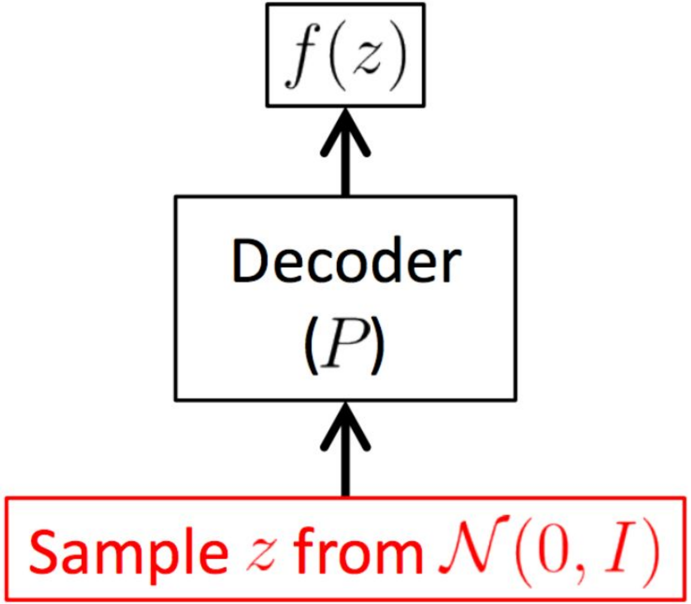
Inference

During inference, decoder Q is discarded and latent zs are sampled from prior distribution of z.

Decoder generates grasps by moving through latent space







Grasp Evaluation & Refinement

Pointnet++ model trained to discriminate **successful** from **unsuccessful** grasps

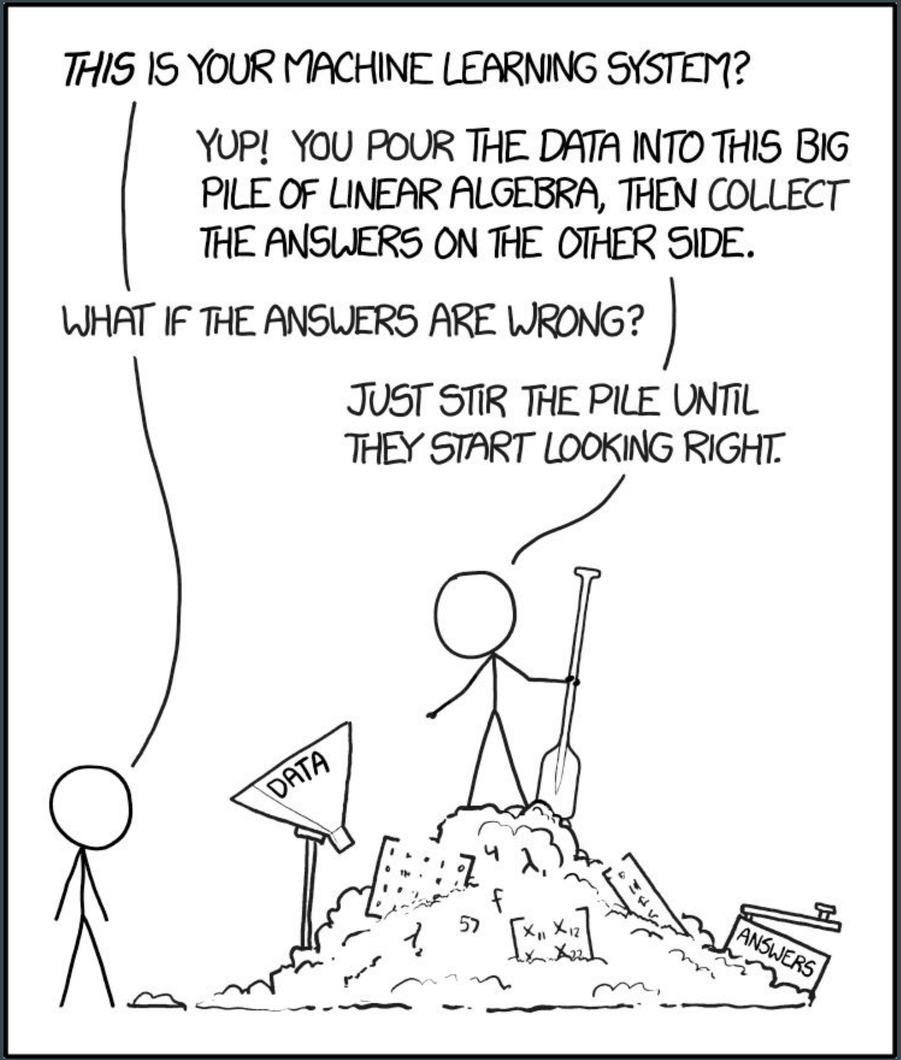
Combined point cloud with binary feature indicating object point or gripper point: captures the relative pose of gripper and object.

Trained as **binary classification** to evaluate the likelihood of **success** for each grasp: $P(S|g, X)$ | **g**: grasp 6D pose, **X**: combined point cloud

Refinement: Search for change in **g** that can maximize improvement in success probability **S** -- use $\partial S / \partial g$

Iterative Refinement!

Machine Learning???



Credit: xkcd.com

Resources & References

[1] Main reference: Stanford Principles of Robot Autonomy-II: Grasping lectures

[2] Friction Cone

[3] Classical Grasp Planning: GraspIt Simulator

[4] Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics

[5] 6-DOF GraspNet: Variational Grasp Generation for Object Manipulation